Revisiting the performance of PCA versus FDA versus Simple Projection for Image Recognition

Fahad B Mostafa

Revisiting the performance of PCA versus FDA versus Simple Projection for Image Recognition

Fahad B. Mostafa

Department of Mathematics and Statistics

The Texas Tech University, Lubbock, USA

Abstract

By the advancement of technology, people are using internet for interchanging millions of photos every day from one to another part of the world, where data reductions are used to send file long distance with in minimum period of time. In the medical science physicians are detecting body organs, tumor cells and complex physical phenomena by optical fibers and where image processing is quite useful. One of another important sector of image processing is meteorology where it processes satellite sending images to do daily weather forecasting or finding climate change. There are many other sectors such as military surveillances, underwater search, satellite navigation etc. Suppose there are millions of images in the database of NSA, but they do not have clear image of the suspect (or suspects). Their main aim is to find the image (or multiple images) of particular scene and identify object of interest in the image (or images). In this study, we will set a very well-known paradigm of analysis using PCA, FDA and simple projection to recognize people from their facial images. We will consider that we have some images of known people that can be used to compare and recognize new images (of the same set of face images). Moreover we will show graphical and tabular representation for average performance of correct recognition as well as analyze the effectiveness of three projections.

Keywords

SVD, orthogonal matrix, orthogonal linear transformation, orthogonal projection, PCA, FDA

Methodology

As images are very high dimensional, it is not easy to analyze them directly. Some common approaches are to reduce their diminution using principal component analysis (PCA), Fisher's discriminant analysis (FDA), and other similar methods. Both PCA and LDA are linear transformation methods. PCA yields the directions (principal components) that maximize the variance of the data, whereas LDA also aims to find the directions that maximize the separation

(or discrimination) between different classes, which can be useful in pattern classification problem such as image recognition.

Our main problem is arising from $Y = X\beta + \varepsilon$ where $Y\ is\ n \times 1$ column matrix, $X\ is\ n \times p$ matrix and ε is $n \times 1$ matrix. If we ignore the error term and compare it with system of linear equation of the standard form $AX = b$ then for $n > p$, the system is overdetermined and we consider it as Linear least square problems (to solve it, we us Gram Schmidt process, QR factorization, Householder, etc), but when $n < p$ then the system becomes underdetermined and we use singular value decomposition shortly SVD to reduce the dimension of column of matrix X from p to smaller dimension (say,'d' for our case). PCA is one of the central uses of SVD.

In our study, the main aim of a PCA analysis is to identify patterns in data; PCA aims to detect the correlation between variables. If a strong correlation between variables exists, the attempt to reduce the dimensionality only makes sense. In a sense PCA is all about finding the directions of maximum variance in high-dimensional data and project it onto a smaller dimensional subspace while retaining most of the information.

Mathematically, PCA is defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by the projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate and do on. Let's say X as,

$$X_{n\times p} = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix}$$

observation

variable

For any matrix $X \in R^{n\times p}$, there exists an orthogonal matrices $U \in R^{n\times n}$ and $V \in R^{p\times p}$ such that $U^T XV = \sum$

Where sigma= $\sum$ = diagonal $(\sigma_1, \sigma_2 \ldots, \sigma_n) \in R^{n\times p}$

$\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \cdots \geq \sigma_n \geq 0$. The $\sigma_i's$ are called the singular values of X and the columns of U and V are called left and right singular vectors of X rexpectively.

For n singular values, $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \cdots \geq \sigma_d \geq \sigma_{d+1} = \sigma_{d+2} \ldots \sigma_n = 0$. Where d is positive singular value, others are zero. Now the rank of our original marix X is d (Figure-1). Range of X is the span of $u_1, u_2, \ldots, u_d$ which is equal to d. Originally we had the dimension of X was $n \times p$ and now it is $n \times d$. Statistically, number of date is reduced from p to d. Now, $U^T XV = \sum$ becomes

$X = U\sum V^T$, that's meaning, in original data we had p columns and now after SVD, we obtain d columns.

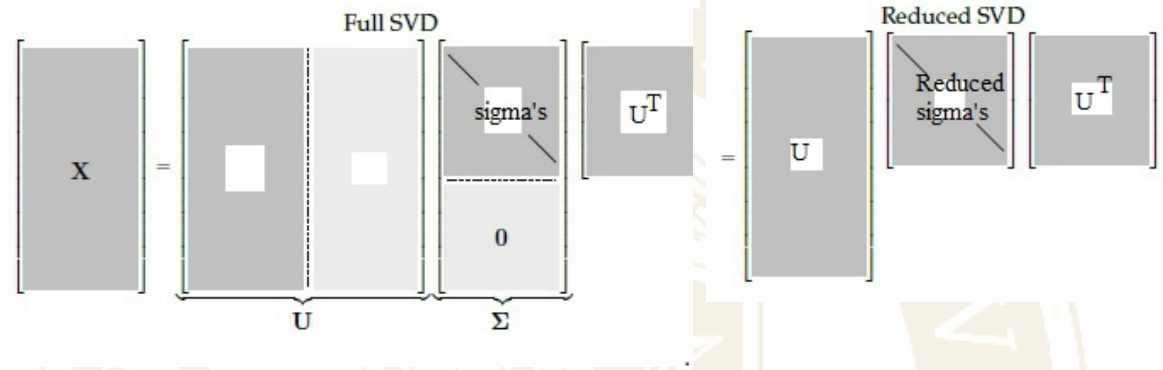

Figure-1: Full singular value decomposition to reduced singular value decomposition

Now we can use principal component analysis (PCA) using this SVD technique. Originally, our X is like a column form

$$X = \begin{pmatrix} X_1 \\ X_1 \\ \vdots \\ X_n \end{pmatrix} \in R^p$$

With mean zero and covariance K. But now it is reduced by $X \in R^d$ which is our principal component. Please note, if X is not 0 then we should use $X - E(X)$ for normalization.

Now the covariance matrix is K which looks like

$$K = \begin{pmatrix} cov(X_1, Y_1) & \cdots & cov(X_1, Y_n) \\ \vdots & \ddots & \vdots \\ cov(X_n, Y_1) & \ldots & cov(X_n, Y_n) \end{pmatrix}$$

Covariance matrix is obtained by $cov(X) = K$.

Suppose U be the $p \times p$ orthogonal matrix such that the elements of the vectors $Z = U^T X$ are uncorrelated. The uncorrelated elements of the vector $Z = U^T X$ are called the component of X. So, our moto is to find this U which can be obtained from the SVD of covariance matrix K. Precisely, $cov(Z) = U^T cov(X) U = U^T K U = \sum$. Now we can compare this SVD.

Thus $Z_i = U_i^T X$ is the principal component. More elaborately $(Z_1, Z_2, \ldots)$ and $(U_1, U_2, \ldots)$ are the principal components and directions respectively. Here, $Z_1$is the largest variance and $Z_2$is the second largest and so on. So, we can reduce the algorithm of PCA of given data as bellows where X denotes an independent observation vector $'X'$. Finding the sample covariance $K \in R^{p \times p}$ then compute the SVD of K to obtain orthogonal matrix $U \in R^{d \times n}$. Then we define principal directions by choosing first column of U. Thus the principal component is

$$Z_{d \times 1} = U_{d \times n}^T X_{n \times 1}$$

Note that, if we have more correlated data then d will be very smaller than n.

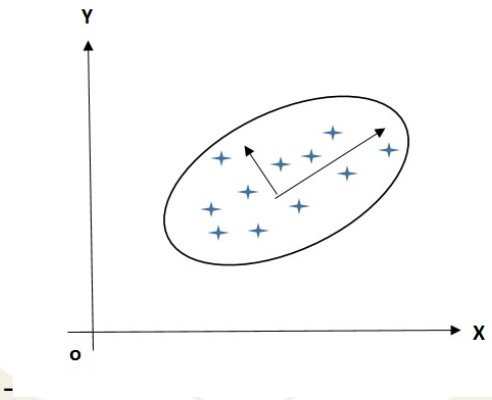

Figure-2: Principal component analysis

One of the most useful demonstration of PCA is the so-called Eigen faces example. There are two sets of images considered for this particular study. One is known as ***training set***; these images are already identified and labeled by some experts, say humans and the second one is ***test images***; these are new images which are need to be identified and labeled. Our aim is to use the similarities between the ***test*** and ***training*** images to label the ***test*** images.

Linear Discriminant Analysis (LDA) is a handful technique for dimensionality reduction technique in the pre-processing step for pattern-classification and machine learning applications. It will

project a dataset onto a lower-dimensional space with good class-reparability in order avoid overfitting ("curse of dimensionality") and also reduce computational costs. Ronald A. Fisher formulated the Linear Discriminant in 1936 (The Use of Multiple Measurements in Taxonomic Problems), and it also has some practical uses as classifier. The original linear discriminant was described for a 2-class problem, and it was then later generalized as "multi-class Linear Discriminant Analysis" or "Multiple Discriminant Analysis" by C. R. Rao in 1948. The general LDA technique is congruent to a PCA, but in addition to finding the component axes that maximize the variance of our data (PCA), we are additionally interested in the axes that maximize the separation between multiple classes (LDA).

In LDA, the goal is to separate and characterize the observation after the projection. Similar observation should be closer and un-similar observation should be separated. In case of multiple class, let us take $C_1, C_2, \dots, C_m$ be m sets that partition. $\hat{R}$ into m classes.

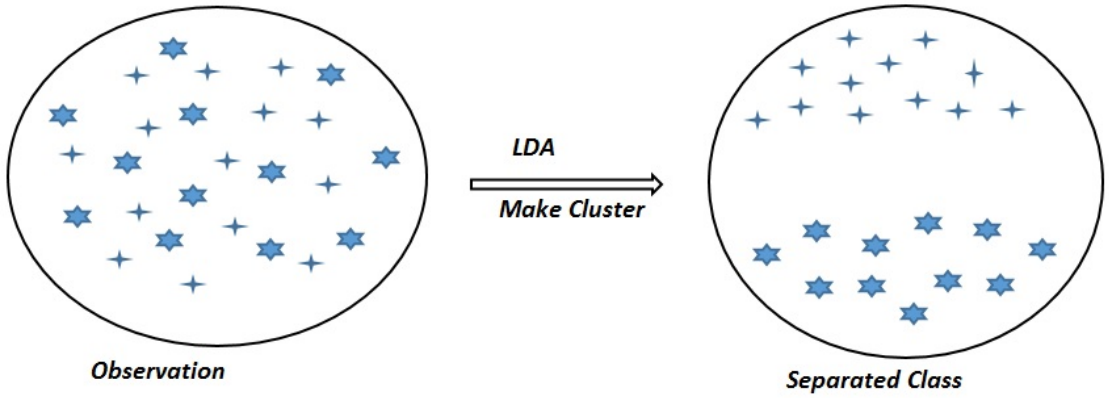

Figure-3: Linear Discriminant Analysis for 2- class

We are given, N_j observations from class labeled by C_j. That is $X_i^j \, \epsilon \, C_j$ for $i = 1,2, \dots, N_j$ & $j = 1,2, \dots, m$

Suppose, we have C_j class

$$C_1 = X_1^1, X_2^1, \dots, X_{N_1}^1 \rightarrow \mu_1 \, \epsilon \, R^n$$

$$C_2 = X_1^2, X_2^2, \dots, X_{N_1}^2 \rightarrow \mu_2 \, \epsilon \, R^n$$

$$\dots$$

$$C_m = X_1^1, X_2^1, \dots, X_{N_1}^1 \rightarrow \mu_m \, \epsilon \, R^n$$

Each $X_i^j \in R^n$, the i^{th} observation in the j^{th} class. Let, μ_j be the mean of observation in class C_j.

$$\mu_j = \frac{1}{N_j}\sum_{i=1}^{N_j} X_i^j \in R^n$$

A matrix captures the separation between the classes in the between class scatter matrix.

$$S_B = \sum_{j=1}^{m}(\mu_j - \mu)(\mu_j - \mu)^T \in R^{n\times n}$$

Where $\mu = \frac{1}{m}\sum_{j=1}^{m} u_j$

A matrix captures the average separate between elements within same class is captured by the within class scatter matrix.

$$S_w = \sum_{j=1,i=1}^{m}\left(\sum_{i=1}^{N_j}(X_i^j - \mu_j)(X_i^j - \mu_j)^T\right) \in R^{n\times n}$$

Now our goal is to find the projection, $Z = U^T X$. The between class scatter matrix because

$$S_B^Z = U^T S_B U$$

The within class scatter matrix becomes

$$S_w^Z = U^T S_w U$$

Now the goal is to choose U that maximizes the following function

$$f(U) = \frac{determinant(S_B^Z)}{determinant(S_w^Z)}$$

$$= \frac{\det(U^T S_B U)}{\det(U^T S_w U)} \begin{matrix}\uparrow\\ \downarrow\end{matrix}; \text{therefore } f(U) \uparrow$$

The optimal projection is $\widehat{U} = argmax\big(f(U)\big); with\ U \in R^{n\times d}$. Now, we have to find U that the quantity $\widehat{U}$ maximize. $\widehat{U}$ can be solved as the generalized eigenvalue problem where we calculate $S_B\widehat{U}_i = e_i S_w \widehat{U}_i$; e_i represents the eigenvalues of the transformation matrix U. If S_B is nonsingular then we can find corresponding eigenvector by calculating $U = {S_B}^{-1} S_w$. In our case we will use building MATLAB code to find eigenvector.

In this study, we compute the principal component (PCA) of training images and project them down to the smaller size. That is, each training image is now represented by a small vector size k. For the test image, we can project them down to a k vector, using the same projection and then we find the nearest image in the training set by computing k vectors. In case of FDA, the low dimensional projection is determined by using scatter matrices.

We will assume n_2 training images each for n_1 people in our database. The size of each image is $s_1 \times s_2$. These images are taken at different orientations, different facial expressions etc.

There are two parts to the procedure. One is to analyze the training images and compute a projection to their principal $k-$dimensional subspace and comparing with training data. In this work we will start from the following.

Let us consider vector in a matrix size $(s_1 \times s_2) \times (n_1 \times n_2)$ is the arrangement of training images and we call it Y_{train}. First n_2 columns are images of person 1, and next n_2 columns are images of person 2, and so on, with a total of $n_1 \times n_2$ columns. Then we use PCA, FDA and Simple projection and we compare these three procedure.

To perform PCA from our dataset Y_{train}, we need to calculate SVD and designate U_1 be the first k columns of the orthogonal matrix U. The size of U is now $(s_1 \times s_2) \times k$.

Now we will use FDA, where we still consider PCA to reduce the size of our data Y_{train} from $(s_1 \times s_2) \times (n_1 \times n_2)$ to $d \times (n_1 \times n_2)$, where $d = \frac{n_1 \times n_2}{2}$. So, we get a new matrix and let us call it $\tilde{Y}_{train}$ and call the $(s_1 \times s_2) \times d$ projection matrix U_0. Use all the vectors from the same person as observations from the same cluster. In $\tilde{Y}_{train}$, the first n_2 columns designate the first person, the second n_2 columns designate the second person, and so on. We have two scatter matric, one is between class and another is with-in class scatter matrices, and use the generalized eigen decomposition to find k eigen vectors that correspond to the largest eigen values. So, we get this result as a matrix form and say it submatrix V. Now we find the orthogonal columns $d \times k$ matrix. Define a $(s_1 \times s_2) \times k$ orthogonal matrix $U_1 = U_0 V$.

We will use simple projection for comparison with PCA and FDA. In this case, we will take another projection where U_1 is simply first k columns of $(s_1 \times s_2)$ identity matrix.

Now, we will repeat the followings for upper three cases one by one. We will use the projection $Y_1 = {U_1}^T Y_{train}$ with size $k \times (n_1 \times n_2)$, which simplify each image in Y_{train} in a reduced form of k −dimensional vector. After using this procedure for each cases we can perform classification. We have n_2 test images per person and the test data set Y_{test} is in the same form as the training data set Y_{train}. We will take an image I randomly from the test set which represents a random column from Y_{test} actually. Projection of I can be found from $I_1 = {U_1}^T I$ where I_1 is a $k \times 1$ vector. Using l_2 norm we can find the distance between I_1 and each column of Y_1. Now, to find the label of the column that has the smallest distance to I_1. If this label matches the true label then our recognition is successful otherwise it is a failure. For computing the percentage of successful recognition $F(k)$, we perform the above procedure 500 times, which will also give us the average performance of upper three methods. After plotting k against each of the three projection we can easily discuss the performance.

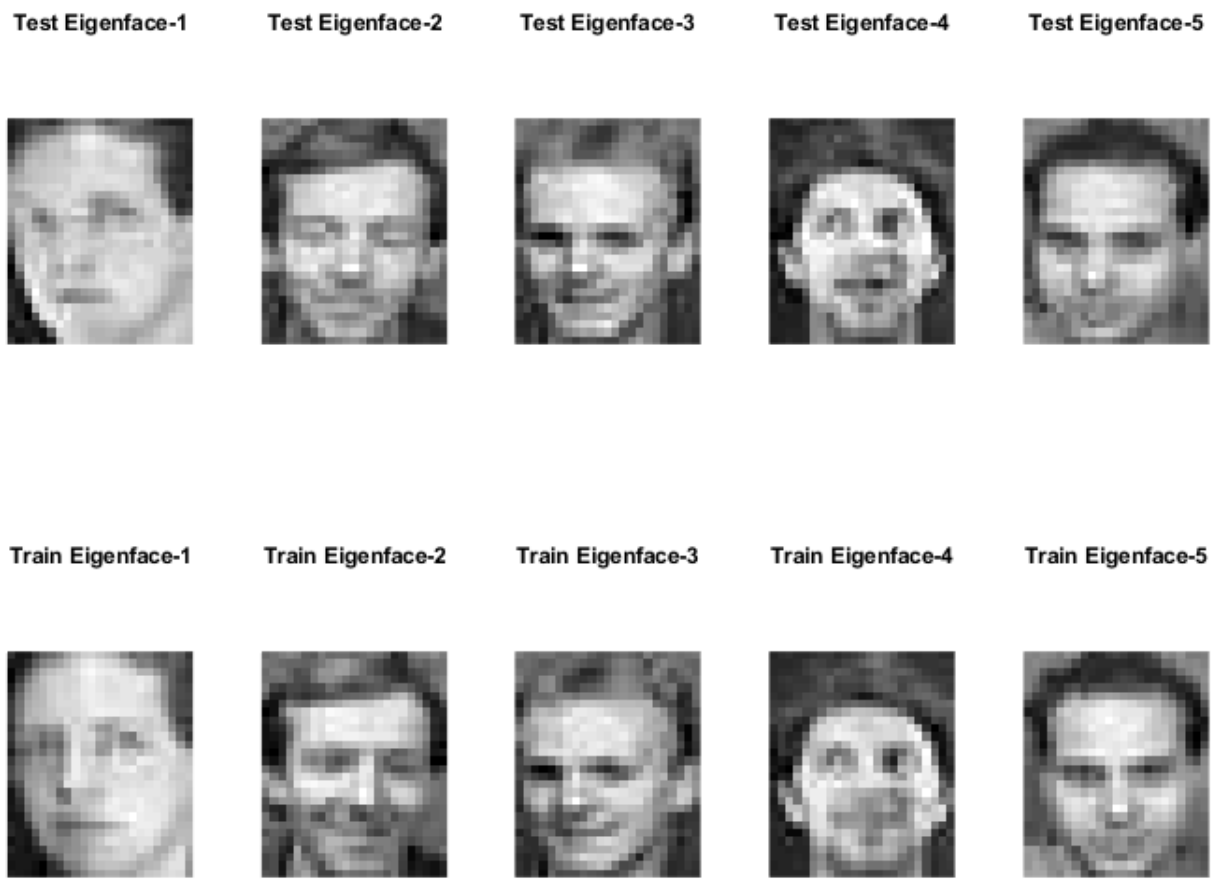

Figure-4: Paradigm of 5 different Eigen faces for test and train datasets

For these image the image size is $s_1 = 28$ and $s_2 = 23$ where $n_1 = 40$ which is the number of people. The number of training images per person is $n_2 = 5$. So we have the matrix size of Y_{train} is $(s_1 \times s_2) \times (n_1 \times n_2)$, i.e. $(28 \times 23) \times (40 \times 5)$. We take 5 different training as well as test images from data set and compare them with each other.

Result discussion:

We have summarized the result by using table and graphs. Then we will show an example of sets of images when the dimensionality parameter k changes then the performance $F(k)$ becomes improved so that the quality of image gets better. We demonstrate it by few examples using test images and the closest images in the training set finally.

Table 1: Representation of average performance of correct recognition $F(k)$ versus the number k for three cases.

k	$F_{PCA}(k)$	$F_{FDA}(k)$	$F_{SP}(k)$	k	$F_{PCA}(k)$	$F_{FDA}(k)$	$F_{SP}(k)$
1	0.114	0.12	0.162	21	0.876	0.836	0.45
2	0.414	0.228	0.19	22	0.872	0.868	0.462
3	0.554	0.438	0.248	23	0.86	0.816	0.464
4	0.608	0.54	0.214	24	0.86	0.862	0.486
5	0.648	0.614	0.246	25	0.862	0.858	0.48
6	0.676	0.622	0.334	26	0.864	0.824	0.46
7	0.752	0.666	0.38	27	0.864	0.872	0.514
8	0.752	0.746	0.398	28	0.882	0.868	0.52
9	0.81	0.782	0.41	29	0.896	0.892	0.566
10	0.846	0.778	0.426	30	0.87	0.9	0.584
11	0.83	0.778	0.408	31	0.872	0.886	0.584
12	0.834	0.792	0.476	32	0.9	0.888	0.578
13	0.818	0.84	0.474	33	0.868	0.876	0.574
14	0.868	0.838	0.504	34	0.884	0.878	0.602
15	0.862	0.85	0.484	35	0.868	0.902	0.572
16	0.832	0.818	0.45	36	0.872	0.912	0.602
17	0.81	0.828	0.474	37	0.854	0.888	0.598
18	0.842	0.856	0.48	38	0.878	0.878	0.534
19	0.844	0.868	0.478	39	0.846	0.894	0.598
20	0.842	0.852	0.484	40	0.888	0.916	0.662

Observation we wish to make from Table-1 is that regarding our training datasets has to do with relative behavior of Simple projection (SP), FDA and PCA as the dimensionality parameter k becomes larger. The performance of these transforms gets better as the value of k increases. What's the difference between the three is that while the recognition rate with PCA saturates around 11.4% to 88.8% when k varies from 1 to 40 while the performance of simple projection vary widely from PCA and FDA. For the experiments under discussion, the average performance of correct recognition best for $F_{PCA}(k) = 90\%$ when $k = 32$, $F_{FDA}(k) = 91.6\%$ when $k = 40$, and $F_{SP}(k) = 66.2\%$ when $k = 40$. From the table we can conclude that the values of $F(k)$ is increasing with the increment of k where FDA shows relatively best performance and SP reflects weaker performance. However the performance of PCA is quite similar to FDA. To understand the result more clearly, we have added a graph of k vs. $F(k)$ below.

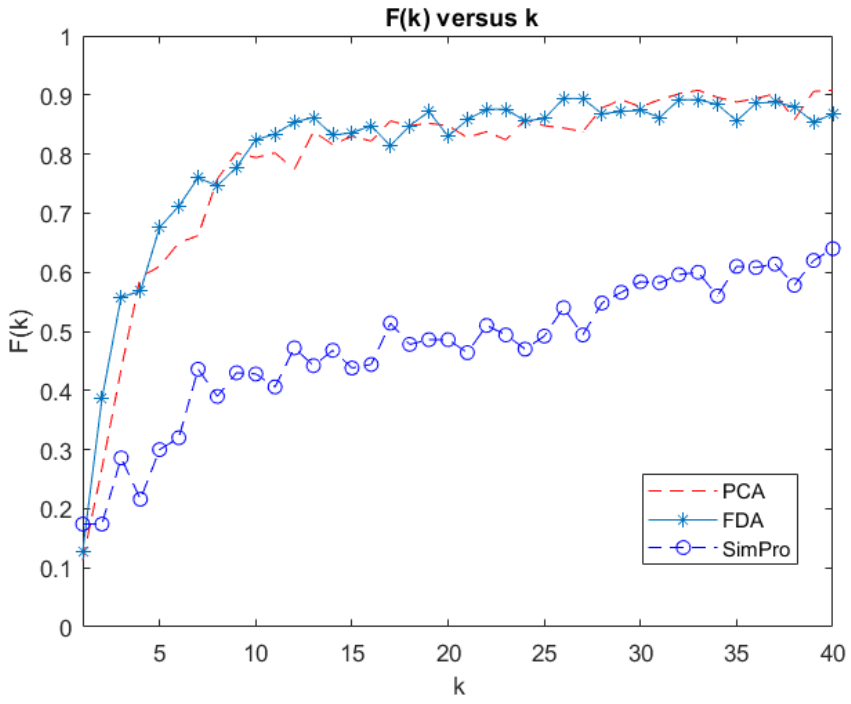

Figure-5: Variation between percentage of successful recognition $F(k)$ and the number k

Here we calculate the average performance. After performing PCA, FDA and Simple projection 500 times and computing the percentage of successful recognition, we draw the graph of k versus $F(k)$. Figure-5 shows LDA is the most proficient technique compared to other two techniques. In fact, LDA > PCA > SimPro in our experiment. The performance of FDA is strictly increasing up to almost 55% when k runs up from 0 to 3 continuously, then there are some

fluctuations of performances with some small ups and downs. FDA achieves almost 91.6% accuracy when dimensionality parameter k becomes 40 which shows best performance in our analysis. In case of PCA, $F(k)$ represents 90% efficiency when k is 32 and then it again declines a little and fluctuates close to 88.8% till k becomes 40. Performance graph of FDA and PCA intersect each other as they both show similar performances. FDA and PCA both shows most of its up grading performance when dimensionality k lays in the interval form 0 to 10 and then they stay in a constant rate almost in the rest of the interval. Hence Simple Projection (SimPro) does not represent good performance compared to other two method. However, Simple projection method also reveals a upward trend of the performance with dimensionality.

To validate our numerical results in table and graphical results with real image, as we want to use our method (PCA, FDA and SP) to recognize new images and compare them with some images of known people. For this case, we have chosen different k values such as $4, 32, 40$ and take a person (say person 2) from test image and compare with our experiment. Figure-6 is indicating the perfection of our study.

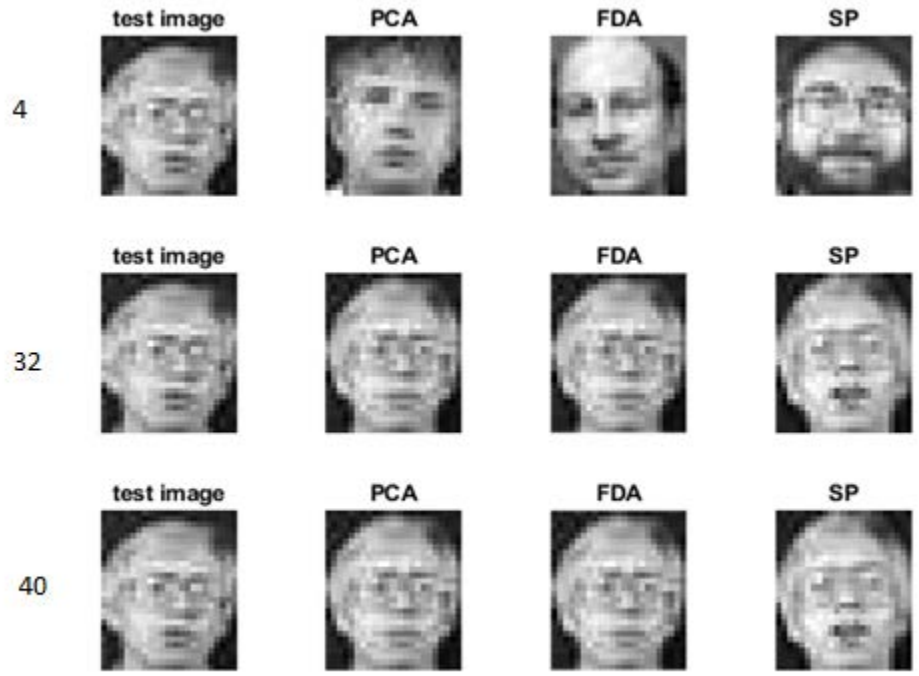

Figure-6: Comparison of Eigen faces using test images with PCA, FDA and SP performed images

Conclusion

In this study, PCA, FDA and simple projection have been executed for face recognition. LDA always outperform PCA since LDA deals directly with class separation, empirical evidence supports our mathematical methods. PCA might outperform LDA when the number of samples per class is small or when the training data non-uniformly sample the underlying distribution. As FDA based on maximizing the ratio between the with-in class and between class variance and this analysis we report validate our claim. That is, FDA is relatively better performing than PCA. However both has good performance under this experiment. In addition, simple projection is easy to compute but which shows quite less performance than PCA and FDA.

Appendix

Used codes

```
close all;
clear all;

load TrainImages.mat;

load TestImages.mat;

% Computing the PCA of Ytrain

[U S V]=svd(Ytrain);   %SVD of Ytrain

[~,Index]=classifier(Ytrain,Ytest); % labeling the testing dataset

    for k=1:40
        tm=0;
        fm=0;
        for m=1:500

            U1=U(:,1:k); % U1 is the 1st k columns of the orthogonal matrix U
            Y1=U1'*Ytrain;  % Ytrain reduced to a k-dimensional vector using
the projection
            %randomly choosing an image from the Ytest
            idx=randsample(200,1);
            I=Ytest(:,idx);
            I1=U1'*I;   % projection of I
            T_index=Index(idx); %index of our desired image in Ytrain data
            [~,ridx]=classifier(Y1,I1); % index of the recognized image
            if ridx==T_index
                tm=tm+1;
            else
                fm=fm+1;
            end
        end
        tm;
        F_PCA(k)=tm/500;
```

```
    end
figure(1);
plot(F_PCA,'r--');
%Computing the FDA of Ytrain

d=200/2;  % define d acording to our problem given

U0=U(:,1:d); % computation of the projection matrix

Ytrain_new=U0'*Ytrain;  % constructing the new training matrix

[V,lambda]=lda(Ytrain_new); %calculation of the matrix V

V=orth(V);  %to make the column orthogonal

for k=1:40
        tm=0;
        fm=0;
        for m=1:500
            V1=V(:,1:k);

            U1=U0*V1; %orthogonal matrix

            Y1=U1'*Ytrain;  %projected reduced Ytrain data

            idx=randsample(200,1);  %randomly selecting image from test data
set
            I=Ytest(:,idx);
            I1=U1'*I;   % projection of I
            T_index=Index(idx); %index of our desired image in Ytrain data
            [~,ridx]=classifier(Y1,I1); % index of the recognized image
            if ridx==T_index
                tm=tm+1;
            else
                fm=fm+1;
            end
        end
        tm;
        F_FDA(k)=tm/500;
end
hold on;
plot(F_FDA,'g');
```

```
function [minval,index]=classifier(Ytr,Yts)
    [k,m]=size(Ytr);
    [l,n]=size(Yts);
    d=zeros(n,m);
    for i=1:n
        for j=1:m
            d(i,j)=(Yts(:,i)-Ytr(:,j))'*(Yts(:,i)-Ytr(:,j));
        end
    end
    [minval,index]=min(d,[],2);
end
```

```
close all;
```

```
clear all;

load TrainImages.mat;

load TestImages.mat;
X=Ytrain; % consider it

[~,Index]=classifier(Ytrain,Ytest); % get the labels of the testing set

subplot(2,5,1)
%reshaping the images
I=reshape(Ytest(:,1),28,23);
%constructing the images
figure(1);
imagesc(I);
colormap gray;
axis equal off;
title ('Test Eigenface-1');

subplot(2,5,2)
%reshaping the images
I=reshape(Ytest(:,11),28,23);
%constructing the images
figure(1);
imagesc(I);
colormap gray;
axis equal off;
title ('Test Eigenface-2');

subplot(2,5,3)
%reshaping the images
I=reshape(Ytest(:,21),28,23);
%constructing the images
figure(1);
imagesc(I);
colormap gray;
axis equal off;
title ('Test Eigenface-3');

subplot(2,5,4)
%reshaping the images
I=reshape(Ytest(:,31),28,23);
%constructing the images
figure(1);
imagesc(I);
colormap gray;
axis equal off;
title ('Test Eigenface-4');

subplot(2,5,5)
%reshaping the images
I=reshape(Ytest(:,41),28,23);
%constructing the images
figure(1);
imagesc(I);
colormap gray;
axis equal off;
```

```
title ('Test Eigenface-5')
%%%%%%%%%%% Ytrain Images %%%%%%%%%%%%%%%%%%%%%%

subplot(2,5,6)
%reshaping the images
I=reshape(Ytrain(:,Index(1)),28,23);
%constructing the images
figure(1);
imagesc(I);
colormap gray;
axis equal off;
title ('Train Eigenface-1');

subplot(2,5,7)
%reshaping the images
I=reshape(Ytrain(:,Index(11)),28,23);
%constructing the images
figure(1);
imagesc(I);
colormap gray;
axis equal off;
title ('Train Eigenface-2');

subplot(2,5,8)
%reshaping the images
I=reshape(Ytrain(:,Index(21)),28,23);
%constructing the images
figure(1);
imagesc(I);
colormap gray;
axis equal off;
title ('Train Eigenface-3');

subplot(2,5,9)
%reshaping the images
I=reshape(Ytrain(:,Index(31)),28,23);
%constructing the images
figure(1);
imagesc(I);
colormap gray;
axis equal off;
title ('Train Eigenface-4');

subplot(2,5,10)
%reshaping the images
I=reshape(Ytrain(:,Index(41)),28,23);
%constructing the images
figure(1);
imagesc(I);
colormap gray;
axis equal off;
title ('Train Eigenface-5');
```

```
subplot(3,4,1)
%reshaping the images
I=reshape(Ytest(:,6),28,23);
%constructing the images
figure(1);
```

```
imagesc(I);
colormap gray;
axis equal off;
title ('test image');

subplot(3,4,2)
%reshaping the images
I=reshape(Ytrain(:,73),28,23);
%constructing the images
figure(1);
imagesc(I);
colormap gray;
axis equal off;
title ('PCA');

subplot(3,4,3)
%reshaping the images
I=reshape(Ytrain(:,63),28,23);
%constructing the images
figure(1);
imagesc(I);
colormap gray;
axis equal off;
title ('FDA');

subplot(3,4,4)
%reshaping the images
I=reshape(Ytrain(:,137),28,23);
%constructing the images
figure(1);
imagesc(I);
colormap gray;
axis equal off;
title ('SP');

subplot(3,4,5)
%reshaping the images
I=reshape(Ytest(:,6),28,23);
%constructing the images
figure(1);
imagesc(I);
colormap gray;
axis equal off;
title ('test image');

subplot(3,4,6)
%reshaping the images
I=reshape(Ytrain(:,7),28,23);
%constructing the images
figure(1);
imagesc(I);
colormap gray;
axis equal off;
title ('PCA');

subplot(3,4,7)
```

```
%reshaping the images
I=reshape(Ytrain(:,7),28,23);
%constructing the images
figure(1);
imagesc(I);
colormap gray;
axis equal off;
title ('FDA');
subplot(3,4,8)
%reshaping the images
I=reshape(Ytrain(:,10),28,23);
%constructing the images
figure(1);
imagesc(I);
colormap gray;
axis equal off;
title ('SP');
subplot(3,4,9)
%reshaping the images
I=reshape(Ytest(:,6),28,23);
%constructing the images
figure(1);
imagesc(I);
colormap gray;
axis equal off;
title ('test image');

subplot(3,4,10)
%reshaping the images
I=reshape(Ytrain(:,7),28,23);
%constructing the images
figure(1);
imagesc(I);
colormap gray;
axis equal off;
title ('PCA');
subplot(3,4,11)
%reshaping the images
I=reshape(Ytrain(:,7),28,23);
%constructing the images
figure(1);
imagesc(I);
colormap gray;
axis equal off;
title ('FDA');
subplot(3,4,12)
%reshaping the images
I=reshape(Ytrain(:,10),28,23);
%constructing the images
figure(1);
imagesc(I);
colormap gray;
axis equal off;
title ('SP');
```

End